EvoEvo Deliverable 2.8

Guillaume Beslon, Charles Rocabert

EvoEvo Deliverable 2.8

Integrated evolutionary model

Due date:	M18 (May 2015)
Person in charge:	Guillaume Beslon
Partner in charge:	INRIA
Workpackage:	WP2 (Development of an integrated modelling platform)
Deliverable description:	Integrated evolutionary model: Documented runnable integrated evolutionary model suitable for running in silico experiments. This model should follow the choices presented in deliverable D2.7.

Revisions:

Revision no.	Revision description	Date	Person in charge
1.0	First version of the integrated evolutionary model	05/05/15	C. Rocabert (INRIA)
1.1	Additions and corrections by G. Beslon and C. Rocabert	22/05/15	C. Rocabert (INRIA)

Table of Contents

1. Introduction

The deliverable D2.8 (*integrated evolutionary model*) is available on the EvoEvo project website (http://www.evoevo.eu/integrated-evolutionary-model/).[1]

This software follows the choices presented in deliverable 2.7 (http://www.evoevo.eu/deliverable-2-7-specifications-of-the-integrated-evolutionary-model/). Some new features, that have not been specified in the deliverable are also included in the code in order to prepare the development of the following models of the project (e.g. membrane permeability or transcriptional noise). Installation instructions are detailed in section 2 and in the file `INSTALL`, in the root of the project. A demonstration parameters file is available in the folder `examples`. The file `README` describes the steps for a first usage of the software (also in section 3). Parameters usage is detailed in `doc/parameters_description.html`. During the simulation, one can have a closer look to the evolution of the system by opening the viewer (`/experiment_path/viewer/viewer.html`) in an internet browser.

Software documentation is available online, among which a « getting started » document, the source code documentation (in Doxygen format) and the description of simulation parameters, all available at http://www.evoevo.eu/integrated-evolutionary-model/.

The software is distributed under the open source GNU General Public License (http://www.gnu.org/licenses/gpl-3.0.en.html).

1.1. The model

The *integrated evolutionary model* includes all the features described in the deliverable 2.7 of the EvoEvo project (http://www.evoevo.eu/deliverable-2-7-specifications-of-the-integrated-evolutionary-model/):

(A) Each cell owns a circular single strand genome made of elements of 5 types: elements coding for enzymes catalyzing metabolic reactions (type E), elements coding for transcription factors (type TF), elements coding for binding sites (type BS), elements coding for promoters (type P), and non coding elements (type NC). At replication, the genome undergoes point mutations, but also large chromosomic rearrangements: duplications, large deletions, inversions, and translocations. The various types of mutation can modify existing genes, but also create new genes, delete some existing genes, modify the length of the intergenic regions, modify gene order...

(B) Each cell owns a genetic regulation network, regulating the evolution of transcription factor and enzyme concentrations during the life of the cell,

(C) Each cell also owns a metabolic network, defined by the whole set of enzymes produces by the cell. Some enzymes act as inflowing or outflowing pumps,

[1] Note that this document is an information note but that it does not constitute the deliverable per se. As specified in the Description of Work document, the deliverable is the documented code.

(D) A dynamic environment is displayed on a two dimensional toroidal grid, on which cells grow. The environment contains various metabolites at various concentrations, which diffuse and are degraded. Cells interact with the environment by pumping metabolites in or out, or by releasing their content at death. In the current version, the environment only contains metabolites (cells cannot exchange DNA or proteins).

1.2. Source package

The source package is displayed as following:

- The `src` folder contains the source code of the project,
- The `examples` folder contains a typical parameters file (default name: `parameters.txt`) used to create a new simulation. Parameters are described in `doc/parameters_description.html`,
- The `doc` folder contains documentation on the *integrated evolutionary model*,
- The `build` folder contains compilation products (binary executables being in `build/bin` folder),
- `Makefile` can be use to compile the project by hand,
- Various other files (LICENSE, AUTHORS, README…).

1.3. License

The software is distributed under the open source GNU General Public License (see http://www.gnu.org/licenses/gpl-3.0.en.html).

1.4. Download

Release(s) of the *integrated evolutionary model* can be downloaded at http://www.evoevo.eu/softwares/.

1.5. Contact

For any question about the software, do not hesitate to contact us via http://www.evoevo.eu/contact-us/ or charles.rocabert@inria.fr.

2. Installation instructions

2.1. Requirements

- Zlib,
- GSL,
- CBLAS,
- SFML 2,
- TBB,
- R,
- R packages: ape, gplots, RcolorBrewer, igraph (and dependencies).

2.2. Supported platforms

The program has been successfully tested on Ubuntu 12.04 LTS, OSX 10.9.5 (Maverick) and OSX 10.10.1 (Yosemite).

2.3. Installation instructions

To compile the project, run the following instructions on the command line:

```
$ cd /path/to/project/
$ make <build_type>, the build types being debug or release
```

Binary executable files are in `build/bin` folder.

3. Typical usage

For a first usage, please take the following steps.

(A) First, place yourself in the `examples` folder:

```
$ cd /path/to/project/examples
```

(B) Create a fresh simulation with the parameters file (`parameters.txt`):

```
$ ../build/bin/create
```

Several folders have been created. They mainly contain simulation backups (population, environment, trees, parameters, …).

Parameters meaning is detailed in `doc/parameters_description.html`.

(C) Alternatively to the `create` executable, use a bootstrap to find a simulation with good initial properties from the parameters file:

```
$ ../build/bin/bootstrap
```

A fresh simulation will be automatically created if a suitable seed is found.

(D) Run the simulation:

```
$ ../build/bin/run –b 0 –t 10000 –g
```

with `–b` the date of the backup, here 0 (fresh simulation), `-t` the simulation time, here 10000 time steps, `-g` displays the graphic window. At any moment during the simulation, one can take a closer look at the simulation by opening `viewer/viewer.html` in an internet browser.

For more information, run any executable with the `–h` option (e.g. `create –h`) and ultimately visit www.evoevo.eu.

4. Executables description

4.1. `create` executable

Create a fresh simulation from a parameters file.

Usage:

```
$ create –h or --help
```

or

```
$ create [-f param-file] [options]
```

Options are:

- `-h, --help:` print this help, then exit
- `-v, --version:` print the current version, then exit
- `-f, --file:` specify parameters file (default: `parameters.txt`)
- `-rs, --random-seed:` specify if the prng seed should be at random

Be aware that creating a fresh simulation in a folder erases previous folders.

4.2. `bootstrap` executable

Find a viable initial population for a given parameters file, via bootstrapping.

Usage:

```
$ bootstrap –h or --help
```

or

```
$ bootstrap [-f param-file] [-min minimum-time] [-pop minimum-pop-
  size] [-t trials] [options]
```

Options are:

- `-h, --help:` print this help, then exit
- `-v, --version:` print the current version, then exit
- `-f, --file:` specify parameters file (default: `parameters.txt`)
- `-min, --minimum-time:` specify the minimum time the new population must survive (default: 100)
- `-pop, --minimum-pop-size:` specify the minimum size the new population must maintain (default: 500)
- `-t, --trials:` specify the number of trials
- `-g, --graphics:` activate graphic display

4.3. `run` executable

Run a simulation from a backup.

Usage:

```
$ run –h or --help
```

or

```
$ run [-b backup-time] [-t simulation-time] [options]
```

Options are:

- `-h, --help:` print this help, then exit
- `-v, --version:` print the current version, then exit
- `-b, --backup-time:` set the date of the backup to load (default: 0)
- `-t, --simulation-time:` set the duration of the simulation (default: 10000)
- `-g, --graphics:` activate graphic display

Statistic files content is automatically managed when a simulation is reloaded from a backup to avoid data loss.

4.4. `experiment` executable

Generate an experiment, containing repetitions on a specified parameters file.

Usage:

```
$ experiment –h or --help
```

Or

```
$ experiment [-f param-file] [options]
```

Options are:

- `-h, --help:` print this help, then exit
- `-v, --version:` print the current version, then exit
- `-f, --file:` specify parameters file (default: `parameters.txt`)
- `-rep, --repetitions:` specify the number of repetitions
- `-t, --simulation-time:` set the duration of the simulation (default: 10000)
- `-rs, --random-seed:` specify if the prng seed should be at random for each repetition

A simulation will be created in a specific folder for each repetition. Bash scripts are also created to create, bootstrap or run the simulations.

4.5. `unitary_tests` executable

Run unitary tests.

Usage:

```
$ unitary_tests –h or --help
```

or

```
$ unitary_tests [-f param-file]
```

Options are:

- `-h, --help:` print this help, then exit
- `-v, --version:` print the current version, then exit
- `-f, --file:` specify parameters file (default: `parameters.txt`)